PATENT
Attorney Docket No. 200352

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

PARKES et al.

Group Art Unit: 2127

Application No. 09/436,618

Examiner: Ali, Syed J.

Filed: November 9, 1999

For: A METHOD OF PIPELINED PROCESSING OF PROGRAM DATA

AMENDMENTS TO THE CLAIMS MADE IN RESPONSE TO OFFICE ACTION DATED FEBRUARY 12, 2003

IN THE CLAIMS:

Please amend claims 1, 3, 8, 15, 23, 26, 30, and 34 as follows:

1. (Amended) A method of carrying out a procedure on a computer system having a memory, the memory containing user context data and global data, comprising: executing a first server, wherein the first server defines a computer-executable function for performing a first sub-task of the procedure; manipulating the global data to carry out the first sub-task; sending the user context data to a second server; executing the second server, wherein the second server defines a computer-executable function for performing a second sub-task of the procedure; and manipulating the global data to carry out the second sub-task using the user context data, wherein the first and second servers are optimized to execute in ~~the~~ cache.

3. (Amended) The method of claim 2, wherein the work packet contains a reply state, and the method further comprises: causing the second server to update the work packet by replacing ~~the~~ a value contained in the action code with ~~the~~ a value contained in the reply state; and causing the second server to send the updated work packet back to the first server.

8. (Amended) The method of claim 7, further comprising: in response to receiving a first work packet containing the user context data; causing the first server to

determine if a node is in the cache; and if the node is determined not to be in the cache, sending a second work packet containing the user context data from the first server to the second server; causing the second server to retrieve the node from a main memory using the second work packet and store the node in the cache; causing the second server to store a reference to the cached node in the second work packet; and sending the second work packet from the second server to the first server, wherein the first server responds to the receipt of the second work packet by searching the cached node.

15. (Amended) The method of claim 13, further comprising defining a reply state code for the work packet, the reply state code being useable by a server to gain information about ~~the~~ results of a function executed by another server.

23. (Amended) A computer-readable medium having stored thereon a data structure, the data structure comprising: a work packet for transferring user context information between at least two servers, wherein each server defines at least one function for performing a sub-task of a computer-executable procedure to manipulate a global data set.

26. (Amended) A computer-readable medium having stored thereon a data structure, the data structure comprising: a first server defining at least one function for performing a sub-task of a computer-executable procedure to manipulate a global data set, wherein the first server executes the function in response to the receipt of a first work packet, the first work packet containing user context information usable by the first server to perform the sub-task, and wherein the first server transmits the user context information to a second server using a second work packet.

30. (Amended) The computer-readable medium of claim 29, wherein the work packet contains a reply state, and the computer-readable medium has further computer-executable instructions for: causing the second server to update the work packet by replacing ~~the~~

a value contained in the action code with ~~the~~ a value contained in the reply state; and causing the second server to send the updated work packet back to the first server.

34. (Amended) The computer-readable medium of claim 33 having further computer-executable instructions for: in response to receiving a first work packet containing the user context data; causing the first server to determine if a node is in the cache; and if the node is determined not to be in the cache, sending a second work packet containing the user context data from the first server to the second server; causing the second server to retrieve the node from a main memory using the second work packet and store the node in the cache; causing the second server to store a reference to the cached node in the second work packet; and sending the second work packet from the second server to the first server, wherein the first server searches the cached node.